REMARKS

In the August 12, 2003, Office Action in this patent application, the United States Patent and Trademark Office (hereinafter "the Office") reopened prosecution. The Office rejected Claims 1-4 and 6-20 under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of U.S. Patent No. 6,237,043, issued to Brown et al. (hereinafter "Brown et al."), taken in view of the teachings of U.S. Patent No. 5,794,004, issued to Lindholm et al. (hereinafter "Lindholm et al."). Claims 1, 17, and 19 were further rejected under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of Lindholm et al., taken in view of the teachings of U.S. Patent No. 6,374,286, issued to Gee et al. (hereinafter "Gee et al."). Furthermore, applicant appreciates the allowability of Claim 5 if it were rewritten in independent form to include limitations of the base claims and intervening claims. The Kishimoto reference (U.S. Patent No. 5,687,073) has been withdrawn.

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art as indicated by MPEP § 2143.03. The cited and applied references do not teach, on the one hand, the concept of using a variable of an object as a pointer to a lock, or, on the other hand, the concept of returning an unused lock to the pool of locks without having to destroy an object previously associated with a lock as recited in Claims 1, 11, 17, and 19. Moreover, a number of cited and applied references cannot be combined, such as Brown et al. and Lindholm et al., without destroying the operation of either reference.

Prior to discussing in detail why applicant believes that all of the claims in this application are allowable, a brief description of applicant's invention and a brief description of the teachings of the cited and applied references are provided. The following background and the discussions of the disclosed embodiments of applicant's invention and the teachings in the cited and applied references are not provided to define the scope or interpretation of any of the

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

claims of this application. Instead, such discussions are provided to help the Office better appreciate important claim distinctions discussed thereafter.

Summary of the Invention

Applicant's invention is directed to avoid or reduce the unbounded creation of lock objects by a recycling process. More specifically, the system of the disclosed invention includes a pool of locks. The pool of locks has a set number of lock objects. By prescribing the use of lock objects from the pool of locks, the unbounded creation of lock objects is inhibited. One or more objects exist in the system of the disclosed invention, and at least one object represents a resource, such as a piece of data or a piece of hardware, that is needed by a number of threads. Each object that represents a resource has a variable, which can act to point to a lock in the pool of locks. The variable of the resource comprises a set of high bits defining the pointer to a lock and a set of low bits defining a status variable. The mechanism that causes the variable of an object to point to a lock, when a thread needs to access a resource represented by the object, is the recyclable locking mechanism of the disclosed invention. The lock returns to the pool of locks when the thread no longer needs to access the resource. The returning lock is in essence "recycled" for the use of another object, thereby conserving computing resources dedicated to creating or maintaining lock objects. Note that the lock is returned to the pool of locks whether or not the object continues to persist in the system of the disclosed invention.

Summary of Lindholm et al.

As spelled out by the Field of the Invention, Lindholm et al. describes his invention as follows:

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

> The present invention relates generally to systems and methods used in object-oriented multithreaded environments for synchronizing objects with threads of execution. In particular, it relates to an object synchronization module and associated method that use a cache of monitors for synchronizing objects with execution threads in a multithreaded environment. (emphasis provided)

Id. Col. 1, lines 5-12.

More specifically, the system of Lindholm et al. is directed to caching and allocating thread synchronization constructs. To synchronize between a plurality of threads of execution and a set of objects, the system of Lindholm et al. provides an object synchronization module, which comprises a cache of synchronization constructs, a hash table containing a list of pointers pointing to allocated synchronization constructs, a list of free synchronization constructs, and a cache manager. When a requesting thread seeks synchronization with an object, the cache manager allocates a synchronization construct in the list for synchronizing the requesting thread with the object only when there is a free synchronizing construct in the list. During the allocation of the synchronization construct, a pointer in the list of pointers is caused to point to the allocated synchronization construct. Moreover, each synchronization construct contains an object identifier, which holds the address of the object being synchronized by the corresponding synchronization construct. If there are no synchronizing constructs (in other words, all of them have been allocated for synchronization among other threads and objects), the requesting thread is placed in a waiting list of the requested synchronization construct. The cache manager de-allocates a synchronization and returns the synchronization construct to the free list by causing the pointer that points to the de-allocated synchronization to point to the free list.

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

Summary of Brown et al.

The system of Brown et al. is directed to adding synchronization capability to an object at run time via a locking mechanism, which is bound to the object by allowing the object to contain a pointer to the locking mechanism. The locking mechanism is bound to the object for the life of the object.

Unlike Lindholm et al., Brown et al. decries the use of monitors for synchronization. Brown et al. explains that a monitor is logically associated with an object. However, in the prior art, a monitor is not bound to the object. Brown et al. disparages the use of monitors as shown in the following text:

> While the use of monitors prevents race conditions from occurring, this approach can significantly degrade the performance of the information handling system. The use of monitors is a time-consuming process. . . . Another problem with the use of monitors is that a monitor is effectively a wrapper around an operating system semaphore. The use of an operating system semaphore requires calls to the operating system, which significantly impacts the performance of the process which is executing. In addition, the monitor's structure contains information which is redundant with the operating system semaphore, such as the owning thread and recursion count. Maintaining this information in tuning two data structures is unnecessary and adds additional overhead to the system.

Id. at Col. 2, line 38 through Col. 3, line 34.

To overcome this problem with the prior art, Brown et al. provides a locking mechanism that is bound to an object for the life of the object.

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

Summary of Gee et al.

The system of Gee et al. is directed to the operation of multiple Java Virtual Machines on a single processor with each Java Virtual Machine operating in a separate time slice called a partition. Each Java Virtual Machine has its own data and control structures and is assigned a fixed area of memory. Each partition is also allotted a fixed period of time in which to operate, and, at the end of the allotted time, a context switch is forced to another Java Virtual Machine operating in the next partition.

The basic storage unit for the processor running the Java Virtual Machines is the object. All objects, as pointed out by the Office, have a "lock control block pointer" (LCB_Ptr) entry. This pointer locates a lock control block which identifies the state of a lock on that object. Lock control blocks are only needed for those objects which are locked at some time during their existence. For those objects that are never locked, the LCB_Ptr is null.

The LCB_Ptr is also null when an object is first created. At the first need for synchronization, a lock control block is allocated from memory. If a lock control block is successfully allocated, the lock control block count field is incremented, indicating the object is locked, and pointers to the object and the current thread control block are added. The count field (designated as reference 902) indicates the number of times the object has been locked. See Gee et al. at Col. 19, Lines 6-7 and 31-36.

The Claimed Invention Distinguised

The Office has failed to show, and applicant is unable to find, where any of the cited and applied references, either alone or in combination, disclose the subject matter of the claimed invention. The present invention is directed to avoid or reduce the unbounded creation of lock

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

objects by a recycling process by prescribing the use of lock objects from the pool of locks, the unbounded creation of lock objects being inhibited. It is accomplished by a system that comprises at least one thread, a pool of locks, at least one object (with a variable) that is capable of representing a resource needed by a thread, and a recyclable locking mechanism. The recyclable locking mechanism associates a lock from the pool of locks with an object using the variable of the object as a pointer when requested by the thread. The lock returns to the pool of locks without having to destroy the object when the thread no longer needs to access the resource. The returning lock is in essence "recycled" for the use of another object, thereby conserving computing resources dedicated to creating or maintaining lock objects. The lock is recycled whether or not the object, which was previously associated with the lock, continues to persist in the system of the disclosed invention.

Using Claim 1 as an example, there is no teaching or suggestion in the cited references for inhibiting the unbounded creation of lock objects, in the manner recited in Claim 1. Claim 1 succinctly defines the system that avoids or reduces the unbounded creation of lock objects by a recycling process. Claim 1 recites a recyclable locking mechanism for associating a lock from the pool of locks with one object using the variable of the object as a pointer when requested by a thread. Moreover, Claim 1 recites that the lock returns to the pool of locks without having to destroy the object when the thread no longer needs to access the resource. The remaining recitations of Claim 1 are directed to the system that allows the lock to be returned to the pool of locks to be in essence "recycled" for the use of another object, hence, conserving computing resources. The cited and applied references do not teach, on the one hand, the concept of using a variable of an object as a pointer to a lock, and, on the other hand, the concept of returning an

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

unused lock to the pool of locks without having to destroy an object previously associated with the lock.

Like applicant's invention, the system of Brown et al. is directed to provide a locking mechanism, which is bound to an object by allowing the object to contain a pointer to the locking mechanism. But the similarity ends there. To understand the differences between the system of Brown et al. and the system of applicant's invention, it should be noted that the term "locking mechanism" as used by Brown et al. may be likened to a lock object of the applicant's invention but cannot be equated to the recyclable locking mechanism. The main problem is that the locking mechanism of Brown et al. remains bound to the object for the life of the object. *See* the Abstract of Brown et al. As further emphasized at Col. 9, lines 52-54, Brown et al. indicates that "once a locking mechanism is bound to an object, it remains assigned to the object for the life of the object." Only when the object is destroyed (such as by garbage collection) will Brown et al. allow another object to use the locking mechanism entry in the locking mechanism table vacated by the destroyed object. In contrast, a lock of applicant's invention returns to the pool of locks whether or not the object that the lock previously secured is destroyed. In other words, a lock object of applicant's invention would return to the pool of locks because a thread no longer needs access to a resource being represented by an object—not because the object was destroyed.

The difficulty with the system of Brown et al. is that unless objects are destroyed, there can be no free locking mechanisms for other threads to use to synchronize access to shared resources. Consider the situation where no object can be destroyed. The system of Brown et al. would permanently stay the execution of multiple threads such that they will have to wait indefinitely to obtain a locking object (which these threads will never in fact obtain) to access

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

shared resources. Not only would this create inadvertent deadlocks, but to solve this problem, Brown et al. would need to provide far more locking objects than can be anticipated. This would be a waste of computing resources.

Unlike applicant's invention, objects in the system of Lindholm et al. lack a variable that can be used as a pointer to point to a lock object and/or be used to indicate the lock status. This minimal design avoids the complexity and the waste of computing resources associated with the use of the object synchronization module of the system of Lindholm et al. It should be noted, however, that each pointer in the list of pointers of the system of Lindholm et al. exists in the context of the object synchronization module to point to a synchronization construct, and therefore, these pointers cannot be likened to the variables of objects of applicant's invention. In fact, objects in the system of Lindholm et al. do not point to allocated synchronization constructs but instead allocated synchronization constructs point to corresponding objects. To accomplish this, the system of Lindholm et al. requires significant overhead, such as a cache of synchronization constructs, a cache manager, a hash table, a hash table address generator, a free list, a cache size controller, and a hash table size controller, in order to manage synchronization constructs so that they can point to corresponding objects. Not so with applicant's invention because synchronization is focused on a simple variable existing in an object that represents a shared resource.

In the Field of the Invention, the system of Lindholm et al. is described as requiring "an object synchronization module and associated method that uses a cache of monitors for synchronizing objects." Brown et al. criticizes the use of a cache of monitors: "While the use of monitors prevents race conditions from occurring, this approach can significantly degrade the

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

performance of the information handling system." *Id.* at Col. 3, lines 10-12. In the abstract, the system of Lindholm et al., unlike Brown et al., does not assign a synchronization construct to an object for the life of the object: "[F]or each specific thread that seeks de-synchronization with a specific object when a specific synchronization construct... is currently located for synchronizing the specific thread with the specific object, the cache manager re-allocates the specific synchronization construct for synchronizing a waiting thread... with the specific object...." The system of Brown et al., on the other hand, indicates that "once a locking mechanism is bound to an object, it remains assigned to the object for the life of the object." *See* Brown et al. at Col 9, lines 52-54.

In its latest Office Action, the Office no longer relies on Lindholm et al. for the teaching of a recyclable locking mechanism for associating a lock from the pool of locks with one object using the variable of the object as a pointer when requested by a thread. But instead, the Office resorts to Brown et al. Brown et al. discusses two different object layouts on the memory heap. FIGURE 3 of Brown et al. focuses on one embodiment of the object layout and FIGURE 4 of Brown et al. focuses on another embodiment. No variable of the object, which acts as a pointer to a lock, is discussed or shown in FIGURE 3 of Brown et al. FIGURE 4 of Brown et al. shows a locking address 130, which points to a locking mechanism stored outside of the object. The problem with Brown et al. is that the locking mechanism is bound to the object for the life of the object.

The Office has indicated that Brown et al. discloses the concept of "returning the lock to the pool of locks when the at least one thread no longer needs to access the resource" by citing the following from Brown et al.:

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

> [T]he "H" bit may be checked. If it is equal to zero, the locking mechanism table entry allocated to the object may be updated to indicate that the table entry is now available for use by another object. This helps minimize the number of locking mechanisms required in the system.

Id. at Col. 9, Lines 40-45. The Office then concluded that Brown et al. does not teach returning the lock to a pool of locks without having to destroy the at least one object.

To understand fully and fairly what Brown et al. actually teaches, applicant reproduces in full the section cited by the Office above:

> Note than [sic] if the object is ever destroyed (e.g., by garbage collection), the "H" bit may be checked. If it is equal to zero, the locking mechanism table entry allocated to the object may be updated to indicate that the table entry is now available for use by another object. This helps minimize the number of locking mechanisms required in the system.

Id. at Col. 9, Lines 40-45 (emphasis provided).

It is not that Brown et al. does not teach returning the lock without having to destroy the at least one object, but that Brown et al. requires an object to be destroyed for the lock to be returned. This is the essence of how the system of Brown et al. works. Brown et al. teaches away from Lindholm et al. and Lindholm et al. teaches away from Brown et al. Lindholm et al. uses a pool of monitors (*id.* at Col. 1, Line 10) and criticizes the employment of a monitor for every object (*id.* at Col. 2, Lines 1-4). Brown et al. bounds a monitor to every object that needs synchronization (*id.* at Col. 3, Lines 61-63) and criticizes the use of a pool of monitors (*id.* at Col. 3, Lines 13-24). Neither Lindholm et al., Brown et al., nor their combination teaches, on the

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

one hand, the concept of using a variable of an object as a pointer to a lock, and, on the other hand, the concept of returning an unused lock to the pool of locks without having to destroy an object previously associated with the lock. The only reference before the Office that teaches these concepts is the pending patent application of the applicant.

To combine, either the approach of Brown et al., which binds a locking mechanism to the object for the life of the object, must be abandoned, or the approach of Lindholm et al., which does not bind a synchronizing construct to the object for the life of the object, must be jettisoned, and the combination would destroy the operation of either reference. Even if the combination of Lindholm et al. and Brown et al. were possible, which applicant specifically denies, these references cannot anticipate or render applicant's invention obvious.

Using Claim 4 as another example, Claim 4 is dependent on Claim 1 and recites that the variable of the object comprises a set of high bits defining the pointer to a lock and a set of low bits defining a status variable. The Office indicated that Brown et al. discloses the use of high bits (locking address 130 at FIGURE 4) and a set of low bits (TakeLock 122 and "H" bit 128 at FIGURE 3). FIGURES 3 and 4 of Brown et al. illustrate two completely different embodiments. FIGURE 3 does not illustrate a variable that can be used as a pointer to a lock. Neither FIGURE 3 nor FIGURE 4 discusses that a set of high bits defines a pointer to a lock and a set of low bits defines a status variable. If the Office insists that this is not the case, applicant respectfully requests that the Office point out precisely where in Brown et al. that "a set of high bits defining a pointer" and "a set of low bits defining a status variable" is discussed.

Using Claim 11 as yet another example, Claim 11 is an independent claim. This method is recited as comprising asserting an instruction by a thread to lock an object. The method

further recites increasing a variable of the object. The variable, in particular, is recited as having a set of high bits for representing a pointer to a lock and a set of low bits for representing a lock status. The method yet further comprises determining whether the variable is greater than a boundary value so as to allocate the lock. The method as yet further comprises recycling the lock by returning the lock to a pool of locks when the thread no longer needs the object regardless of whether the object persists after the lock returns to the pool of locks.

The Office has indicated that Brown et al. teaches the act of "increasing a variable of the object" by noting that the element TakeLock 122 of Brown et al. can be incremented. The element TakeLock 122 of Brown et al. is illustrated in FIGURE 3, which does not have a variable to point to a lock. FIGURE 3 illustrates an embodiment that is different from the embodiment shown in FIGURE 4. Even if the element TakeLock 122 can be incremented, it is difficult to understand how this could be equated to the act of "increasing a variable of the object." The element TakeLock 122 is not a variable that can point to a lock.

As discussed above, all objects in the system of Gee et al. have a "lock control block pointer" (LCB_Ptr) entry. At the first need for synchronization, a lock control block is allocated from memory. If a lock control block is successfully allocated, the lock control block count field is incremented, indicating the object is locked and pointers to the object and the current thread control block are added. The count field (designated as element 902) indicates the number of times the object has been locked. See Gee et al. at Col. 19, Lines 6-7 and 31-36. A count of zero means the object is unlocked. To repeat for emphasis, the count field is associated with a lock control block. The lock control block is pointed to by the lock control block pointer in the object. In order for the count field to intimately know about the number of times a

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

corresponding object in the system of Gee et al. is locked or unlocked, the lock control block must be associated with the object for the life of the object. Unless the object is destroyed in the system of Gee et al., the corresponding lock control block is not freed. Like Brown et al., Gee et al. does not teach, on the one hand, the concept of using a variable of an object as a pointer to a lock, and, on the other hand, the concept of returning an unused lock to the pool of locks without having to destroy an object previously associated with the lock. To combine, either the approach of Gee et al., which binds a lock control block to the object for the life of the object, must be abandoned, or the approach of Lindholm et al., which does not bind a synchronizing construct to the object for the life of the object, must be jettisoned, and the combination would destroy the operation of either reference. Even if the combination of Lindholm et al. and Gee et al. were possible, which applicant specifically denies, these references cannot anticipate or render applicant's invention obvious.

Because the Office has failed to state a *prima facie* case of obviousness, the rejection should be withdrawn. Independent Claims 1, 11, 17, and 19 are clearly patentably distinguishable over the cited and applied references. Claims 2-10, 12-16, 18, and 20 are allowable because they depend from allowable independent claims and because of the additional limitations added by those claims. Consequently, reconsideration and allowance of Claims 1-20 is respectfully requested.

CONCLUSION

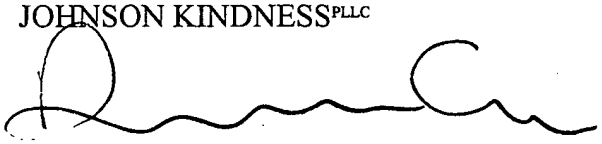
In view of the foregoing remarks, applicant submits that all of the claims in the present application, as amended, are clearly patentably distinguishable over the teachings of Brown et al., Lindholm et al., and Gee et al. taken alone or in combination. Thus, applicant submits that this application is in condition for allowance. Reconsideration and reexamination of the

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100

application, allowance of the claims, and passing of the application to issue at an early date are solicited. If the Examiner has any remaining questions concerning this application, the Examiner is invited to contact the applicant's undersigned attorney at the number below.

Respectfully submitted,

CHRISTENSEN O'CONNOR
JOHNSON KINDNESS[PLLC]



D.C. Peter Chu
Registration No. 41,676
Direct Dial No. 206.695.1636

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the U.S. Postal Service in a sealed envelope as first class mail with postage thereon fully prepaid and addressed to Mail Stop Non-Fee Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the below date.

Date: November 12, 2003 Cindy R. Morton

DPC:clm

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS[PLLC]
1420 Fifth Avenue, Suite 2800
Seattle, Washington 98101
206.682.8100